Here's a comparison of how attributes are used in **JSX** versus **HTML**, highlighting key differences:

---

1. Class Attribute

HTML	JSX
`<div class="container">Hello!</div>`	`<div className="container">Hello!</div>`

- **Why?**: In JSX, `class` is a reserved keyword in JavaScript, so `className` is used instead.

---

2. Inline Styles

HTML	JSX
`<div style="color: red; font-size: 20px;">Hello!</div>`	`<div style={{ color: "red", fontSize: "20px" }}>Hello!</div>`

- **Why?**: In JSX, the `style` attribute accepts an object, with CSS properties written in camelCase and values as strings (for unitless values, numbers are allowed).

---

3. Event Handlers

HTML	JSX
`<button onclick="alert('Clicked!')">Click</button>`	`<button onClick={() => alert('Clicked!')}>Click</button>`

- **Why?**: Event names in JSX are written in camelCase (e.g., `onClick` instead of `onclick`). You can use inline functions or references to JavaScript functions.

---

4. Boolean Attributes

HTML	JSX

`<input type="checkbox" checked>`	`<input type="checkbox" checked={true} />`

- **Why?**: In JSX, boolean attributes are specified explicitly as `true` or omitted for `false`.

---

5. Dynamic Attributes

HTML	JSX
`<img src="/images/pic.jpg" alt="Image">`	`<img src={imageUrl} alt={altText} />`

- **Why?**: In JSX, attributes can accept JavaScript expressions inside `{}` for dynamic content.

---

6. Custom Attributes

HTML	JSX
`<div data-id="123">Hello</div>`	`<div data-id="123">Hello</div>`

- **Why?**: Custom attributes like `data-*` work the same way in both HTML and JSX.

---

7. TabIndex Attribute

HTML	JSX
`<div tabindex="0">Focusable</div>`	`<div tabIndex="0">Focusable</div>`

- **Why?**: JSX uses camelCase (`tabIndex`) for attributes like `tabindex`.

---

8. Self-Closing Tags

HTML	JSX
`<img src="/path/to/image.jpg" alt="Image">`	`<img src="/path/to/image.jpg" alt="Image" />`

- **Why?**: In JSX, self-closing tags must include a trailing `/`.

---

9. For Attribute

HTML	JSX
`<label for="inputId">Label</label>`	`<label htmlFor="inputId">Label</label>`

- **Why?**: `for` is a reserved keyword in JavaScript, so `htmlFor` is used instead.

Summary of Key Differences

Feature	HTML	JSX
Class Attribute	`class="..."`	`className="..."`
Inline Styles	`style="..."`	`style={{ ... }}`
Event Handlers	`onclick="..."`	`onClick={...}`
Boolean Attributes	`checked`	`checked={true}`
Dynamic Values	Static Strings Only	Supports JavaScript `{}`
Self-Closing Tags	Optional `/`	Mandatory `/` for Empty Tags
Reserved Keywords	`for`, `class`, etc.	`htmlFor`, `className`, etc.

By understanding these differences, you can write JSX effectively while leveraging your HTML knowledge.